Project No. 007 001 01
Document No.
TRACOR 67-904-U

AD 661861

DATA MANAGEMENT
A COMPARISON OF SYSTEM FEATURES

T. W. Ziehe
TRACOR, Inc.

October 1967

TRACOR

6500 Tracor Lane, Austin, Texas 78721, AC 512/926-28

43

TRACOR 6500 TRACOR LANE AUSTIN TEXAS 78721

Project No. 007 001 01
Document No.
TRACOR 67-904-U

DATA MANAGEMENT
A COMPARISON OF SYSTEM FEATURES

T. W. Ziehe
TRACOR, Inc.
October 1967

TRACOR 6500 TRACOR LANE AUSTIN TEXAS 78721

TABLE OF CONTENTS

ACKNOWLEDGMENTS

This report represents work under contract N00014-67-C-0395 with the Office of Naval Research, Information Systems Branch. It is based both on publicly available documentation and on personal conversation. The cooperation of all individuals contacted is acknowledged, in particular A. H. Vorhaus and R. Lickhalter at S.D.C., A. G. Dale and J. DeLine at The University of Texas, and J. Sable and H. Stout at Auerbach Corporation. The author was personally involved in the development of the catalog system; other principal contributors are M. Kay and D. G. Hays at RAND.

TRACOR 6500 TRACOR LANE. AUSTIN. TEXAS 78721

ABSTRACT

Features of four data management systems, all being developed, are compared. The four systems are the Time-Shared Data Management System and a variant of it, the Remote File Management System; Data Manager-1; the Generalized Information System; and the Catalog System. Comparisons are limited to two areas: external and internal data structuring and organization. Several differences among the systems are noted and briefly discussed.

TRACOR 6500 TRACOR LANE AUSTIN TEXAS 78721

DATA MANAGEMENT
A COMPARISON OF SYSTEM FEATURES

This paper compares some features of four data management systems being developed:

1. TDMS and RFMS* - the former under development at System Development Corporation, the latter at the Computation Center of The University of Texas at Austin.
2. DM-1 - under development at Auerbach Corporation and Rome Air Development Center.
3. GIS - under development at IBM, Federal Systems Division.
4. Catalogs - under development at The RAND Corporation, Linguistics Research Project.

It assumes a familiarity with the field of data management and some background of knowledge about the systems discussed. The names used above and in following sections represent more descriptive titles:

TDMS - Time-Shared Data Management System
RFMS - Remote File Management System
DM-1 - Data Manager-1
GIS - Generalized Information System

Documents listed in the bibliography are the source of most of the material presented. They also clarify many related issues not discussed in this report. In many cases definitions and rules ascribed to a system are direct quotations taken from

*These two systems are treated together since RFMS is largely a re-implementation of TDMS for the CDC-6600. They differ slightly in the areas discussed here, but in ways that need not be made explicit.

these documents. In some cases editing changes were made, in a few a complete restatement is used. Conversations with individuals working on TDMS, RFMS, and DM-1 clarified several points covered, but if misrepresentations persist, they are the sole responsibility of the author.

Although none of the systems is in full-scale use, each has reached a sufficiently advanced stage that documentation describes the features compared: external and internal data structuring and organization. External refers to the structure imposed by a user of the system on his data; that is to say, his description of datum relationships. Section 1 deals with comparisons in this area. Internal refers to the structure and organization imposed on the data by the system itself. This includes the system's handling of both datum values and information about those values, some of which the user supplies but some of which the system generates. Section 2 contains comparisons in this area.

1. EXTERNAL DATA STRUCTURING AND ORGANIZATION

Each data management system offers its users a capability for describing a datum array which the system is to manage. In this description the user defines data classes and their structural relations. The system stores this information, using it to guide processing of the data.

This capability is of crucial importance. Its design determines the system's naturalness, from the user's point of view, in each application. It also influences the number and complexity of the system's basic processes and contributes to the system's operating efficiency. The four systems show a similarity in their approach to external data structuring and organization. A sketch of this common ground provides a vocabulary and framework for comparison.

Each system offers its users a budget of components with which to construct a structural description. This description or map is a hierarchy of components, related by inter-component connections in a manner consistent with system rules. Given a map, a system is able to construct and deal with a data base that corresponds to it. Each element of the data base, that is, each datum, corresponds to a component of the map and exhibits a pattern of relationships with its neighboring data that matches a pattern of component relationships in the map. System rules regulate the repetition of map patterns in the data base, but in general each component corresponds to many datum occurrences in the data base. Each datum either carries information directly in the form of its value or simply groups subordinate data. Only the catalog system allows a datum to do both simultaneously. Names attached by the user to each component are used to access datum values in the data base.

The four systems are similar in that maps and data bases are tree-shaped. Each component or datum corresponds to a

node of the tree and has only one parent, although multiple offspring are allowed. Components or data with a common parent are called siblings.

In the remaining paragraphs of this section the following features are described for each system: the components, the type of data to which each corresponds, and the facilities provided for attaching external names; the rules that govern the composition of a map; and the rules of correspondence between components in a map and data in an array.

1.1 COMPONENTS, DATUM TYPES, AND EXTERNAL NAMES

The terms used as component names are listed for each system with a description of the datum type to which each corresponds. For value-bearing components a summary description of value encoding is included. External naming conventions for components are also sketched, but additional encoding and naming conventions can be added to any system without affecting the rest of the design.

TDMS and RFMS. Two components are available in these systems:

1. An element corresponds to datum values that can be declared as names (alphanumeric), numbers (real number), dates (month, day, year), or text (to be implemented in a later version). An element has no substructure in the map.

2. A repeating group is a component with a substructure of one or more components, either elements or subordinate repeating groups. A repeating group corresponds to data that group subordinate data.

Each component can be assigned an external name. In addition, TDMS and RFMS offer users two other naming devices. Each component can be assigned a number by the user which is

independent of a component identity number assigned by the system. The second device, an overlay capability, defines a sub-element as a portion of a previously defined element. This adds no structure to the map; it merely provides the user with an alternate data access, normally to part of a value rather than the whole.

DM-1. Maps are constructed from three components (in DM-1 "items" is a synonym for "components"):

1. A field is a terminal item; that is, it has no substructure in the map. It corresponds to value-bearing data and is defined by its name and the type, size, and unit designators for its values. A field name is an alphanumeric string which is used externally. The type specifies the coding scheme used for values of the field. It may be alphanumeric, integer, binary, octal, decimal, or exponential. The size is a measure of the length of field values (alphanumeric characters, binary bits, decimal digits, etc.) and may be fixed or variable length. The units designator specifies the scale on which the value of the field is measured; e.g., volts, amperes, meters, etc.

2. A file is an item with an arbitrary number of sub-items, each with an identical structure, as its substructure. Each sub-item is a record. The file is defined by its name and the structural definition of its records. The data value of a file is the group of values of the fields subsumed by all the records of the file.

3. A statement is an item with fields, files, or other statements in its substructure. A statement is defined by its name and the definitions for its

sub-items. In effect, the statement is a mechanism for associating several related items to show the relationship and permit them to be treated as a unit. The data value of a statement is the set of values of its subsumed fields.

Sections 1.2 and 1.3 will help to clarify the roles of statements and files in DM-1. At this point, it is sufficient to say that both correspond to datum-grouping, non-terminal data while a field component corresponds directly to value-bearing data. The user assigns each component an external name, a variable-length string of alphanumeric characters. No provision is made for referencing portions of field component values as in TDMS.

GIS. Maps are constructed from two components:

1. A field is a terminal GIS item -- the smallest unit of data that can be referenced or described. The required descriptive elements for a field are (1) its relative position in the segment, (2) its name (and synonym, if any), (3) its type, and (4) its length. Field type may be binary, decimal, floating point, or alphanumeric. An alphanumeric field may be either fixed or variable in length, whereas all others must be fixed-length. Length control for a variable length field may be provided by one of two options: count or terminating code. In the count option, each occurrence of the variable length field will be preceded by a "length" field indicating the number of characters for this occurrence of the field. In the terminating-code option a user-selected special character is employed to mark the end of each occurrence. A set of consecutive fields is called a segment.

2. A file is composed of an arbitrary number of items, called records, each with the same logical structure. A GIS record is composed of a single segment followed by zero or more embedded files. The segment is a defined set of fields; the structure of each embedded file follows the rules of file structure.

The GIS file corresponds exactly with the TDMS repeating group and the DM-1 file; the GIS field with the TDMS element and the DM-1 field.

Catalogs. Maps are constructed from a single component, a data class. A class corresponds directly to value-bearing data and is defined by its name and the type of encoding used for its values. Encoding types include text, binary, and strings of alphanumeric characters. Datum values are assumed to be variable length so no indication of length is given with the class. The class name is a sequence of three alphanumeric characters.

1.2 THE MAP OR STRUCTURAL DESCRIPTION

In each system rules govern the construction of a structural description. Thinking of a structural description as two dimensional, these rules prescribe both the hierarchical and sibling relationships which are allowed. These features are described and compared in the following paragraphs.

TDMS and RFMS. The rules regulating the hierarchical relationships in a structural description are:

1. An element (E) may occur either without a parent component or with a repeating group as its parent component. An element is never a parent component.

2. A repeating group (RG) may occur either without a parent component or with a repeating group as its parent component. A repeating group is always the parent of at least one component.

Thus, every structural description is built from these three types of hierarchical relationships:

```
E   RG   RG
    |    |
    E    RG
```

There are no direct restrictions on sibling relationships. A structural description can contain any sequence of components-without-parents as well as any sequence of components as sibling offspring of a repeating group.

Two general restrictions are made on structural descriptions:

1. The number of levels in the hierarchy must not exceed sixteen.
2. The number of components used in the description must not exceed 1023 elements and 255 repeating groups.

DM-1. These rules regulate the hierarchical relationships in a structural description:

1. A field (D) may occur without a parent or with either a statement or file as its parent. A field is never a parent component.
2. A statement (S) may occur without a parent or with either a statement or file as its parent. A statement is always the parent of at least one component.
3. A file (F) is restricted in the same way as a statement.

Thus, every structural description is built from these seven types of hierarchical relationships:

```
D   S   S   S   F   F   F
    |   |   |   |   |   |
    D   S   F   D   S   F
```

There are no restrictions on sibling relationships. A structural description can contain any sequence of components-without-parents as well as any sequence of components as sibling offspring of either a statement or file.

No general restrictions on structural descriptions are mentioned in current documentation.

GIS. These rules regulate the hierarchical relationships in a structural description:

1. A field (D) must occur with a file as its parent component, but a field is never itself a parent.
2. A file (F) may occur either without a parent component or with another file as its parent. A file is always the parent of at least one field.

Thus, every structural description is built from these two types of hierarchical relationships: $\begin{array}{c} F \\ | \\ D \end{array} \quad \begin{array}{c} F \\ | \\ F \end{array}$

Sibling relationships are regulated by the following rule: the offspring of a file component are ordered so that all fields precede any files which occur. In addition, a number of rules affect the co-occurrence of fields and subordinate files in a structural description; for example, records of a subordinate file can be identified either by a record count or key value in a field that is a sibling of the file.

Catalogs. A single rule regulates the hierarchical relationships in a structural description:

1. A class may occur either with or without a parent component. It may also occur as a parent or without offspring.

Thus, every structural description is built from these two types of hierarchical relationships: $\begin{array}{c} C \\ \\ \end{array} \quad \begin{array}{c} C \\ | \\ C \end{array}$

Sibling relationships are not regulated; a class may have any number of offspring classes. A structural description can consist of no more than 512 classes.

From the above it should be noted that the catalog system combines in one component properties that other systems divide among several. This fact is simply noted here but it is the root of an important difference in approach: the catalog system structures the data of an array directly, whereas the other systems group data under a hierarchy of names.

1.3 THE DATUM ARRAY

Each system creates datum arrays which correspond in a prescribed way with a map that has been supplied. Datum patterns in the array correspond to component patterns in the map. System rules define this correspondence, both the repetitive use of patterns from the map in the array and the extent to which parts of the pattern from the map can be omitted in the array.

TDMS and RFMS. The correspondence between map components and data in an array is regulated by these rules:

1. The pattern of the entire structural description can recur any number of times. Each occurrence is called a logical entry of the data base.

2. Within each logical entry each repeating group component can correspond to any number of datum nodes. The substructure of the repeating group is repeated for each occurrence.

3. Each instance of a subordinate repeating group generated by its parent repeating group can itself correspond to any number of datum nodes, repeating its substructure with each.

4. Each instance of an element component generated by the first three rules corresponds to a single datum node.

Each element is an optional participant in each repetition of its parent. An instance of a repeating group is said to exist only if there is at least one element value occurring in its substructure.

DM-1. The correspondence between map components and data in an array is regulated by these rules:

1. Each occurrence of a statement component can correspond to only one datum node. The statement's substructure is the substructure of that datum.

2. Each file component can correspond to one file datum that subsumes any number of datum nodes, each called a record. The substructure of the file is repeated in each record.

3. Each instance of a subordinate file generated by its parent file or statement can itself generate any number of record nodes, repeating its substructure's pattern for each.

4. Each instance of a field component generated by the first three rules corresponds to a single datum node.

Each component is tagged as either required or optional. An optional component plus its substructure can be omitted from any occurrence generated by its parent. Required components must be included in each repetition.

GIS. The correspondence between map components and data in an array is regulated by these rules:

1. Each file component can correspond to one file datum that subsumes any number of record nodes. The pattern of the file's substructure is repeated for each record.

2. Each instance of a subordinate file generated by its parent file can itself generate any number of record nodes, repeating its substructure with each.

3. Each instance of a field component generated by the first two rules corresponds to a single datum node.

No provision is made for optional fields or files. Thus, each repetition of a structural pattern generates data for each field and file in it.

Catalogs. The correspondence between map components and data in an array is regulated by these rules:

1. Each class component can correspond to any number of datum nodes. The substructure of the class is the substructure of each datum.

2. Each instance of a subordinate class generated by its parent class can itself correspond to any number of datum nodes, repeating its substructure with each.

Every class is an optional participant in each repetition of its parent class. When a particular class is not included, all classes in its substructure are also excluded.

2. INTERNAL DATA ORGANIZATION

A system's provision for describing data structures is an avenue of approach to its main body of capabilities for data management. In this section, we examine the base upon which this main body of capabilities rests. It includes the form in which the structural description is stored, the techniques used to represent the prescribed structure within the datum arrays, and the format within each system for physical storage. It also includes any secondary organizations of information developed by the system from the datum arrays. This additional information prepares the system for activity of a particular type or style. This part of the base is a measure of the extent to which a system is committed to processing data in a particular way.

This discussion of system architecture covers the way each system deals with the user's structural description and other information _about_ the data, as well as the way each deals with the user's data itself. At times both kinds of information will be referenced simply by the term _data_. When a distinction must be made, th user's data will be referred to as _descriptive data_, data that describe things to which the system does not have access. Information about the descriptive data, whether provided by the user or generated by the system itself, is referred to as _logical data_, logical with respect to system processes that use it.

The four systems under consideration are quite diverse in the areas referred to above. Because there are only a few similarities, a point by point comparison is impossible; each system is therefore described separately. The documentation available on TDMS and RFMS and on DM-1 makes possible a more extensive description of the architecture of these systems. The information at hand for GIS is much less complete; therefore, its description is brief. The Catalog System offers very limited

capabilities in the area under discussion. Consequently, its description is also brief.

2.1 TDMS and RFMS

These systems enable individuals and organizations with large, and often complex, data files to manage their data with speed and ease. They are oriented to the nonprogrammer user, who, after learning some basic commands, can work directly with the computer to manage his data base. The systems permit the user (1) to describe entries in a data base, (2) to load them into the machine, (3) to ask questions about them, (4) to perform calculations on them, (5) to have the data displayed on a cathode-ray-tube, (6) to obtain hard-copy reports, and (7) to update and maintain the data base.

These objectives influence the internal architecture of both systems. A further influence is the fact that both are designed for on-line operation. Several users can work concurrently, each with his own data base. Each system maintains a list of names of data bases to which it has been introduced, but beyond that each data base is independent of all others. Each data base, both its logical and descriptive data, is organized for storage on a secondary device that offers direct access. The systems block and unblock the data and, together with the operating system, manage the storage and retrieval of blocks. Magnetic tape is used only as a back-up for direct access storage when a data base cannot remain on-line. All active data bases must be stored in an on-line, direct-access device.

2.1.1 Internal Architecture of TDMS and RFMS

Each data base received, both its logical and descriptive data, is organized by and stored as the content of fourteen tables. This complex of tables, shown in schematic form in Figure 1, reveals the internal architecture of TDMS. Each table is represented by its symbolic name except for the tables "legal"

and "units"; they are as yet undefined and unnamed. The set of tables CDEFINA, CDEFINB, CDEFINC, CDEFIND, CDEFINE are parallel in that there is always a one-to-one correspondence among their entries. They could be combined into one table, but are separated for increased efficiency.

Many TDMS/RFMS tables are actually a series of sub-tables. Each table or sub-table is a series of entries and an entry consists of one or more items. An item type is either fixed or variable length and contains either a logical or descriptive datum. The logical data that relate one table or sub-table to another are shown in Figure 1 as arrows. The significance and use of these paths through the complex will become clearer in later paragraphs.

The design of the table, sub-table, entry, item hierarchy is independent of the rules provided the user of TDMS or RFMS for structuring his data. In fact, the available documentation does not formalize the rules which govern this hierarchy. As a consequence of this dual approach there is a clear distinction between extending the system and applying the system to a task. In fact, the tendency will be for different individuals to perform the two tasks. At least appliers of the system will seldom be able to extend the system and its capabilities.

The fourteen TDMS/RFMS tables fall into three functional groups. Figure 2 illustrates this grouping. The tables in the first group contain information about the components of the data base -- each element and repeating group. The content of these tables is essentially the structural description provided by the user. The second group of tables contains a value glossary for the values of each element. They record an organization of the data generated by the system in addition to that provided by the user. The third preserves the datum array as organized by the user's map or structural description. The remaining paragraphs of this section describe each group of tables and their content.

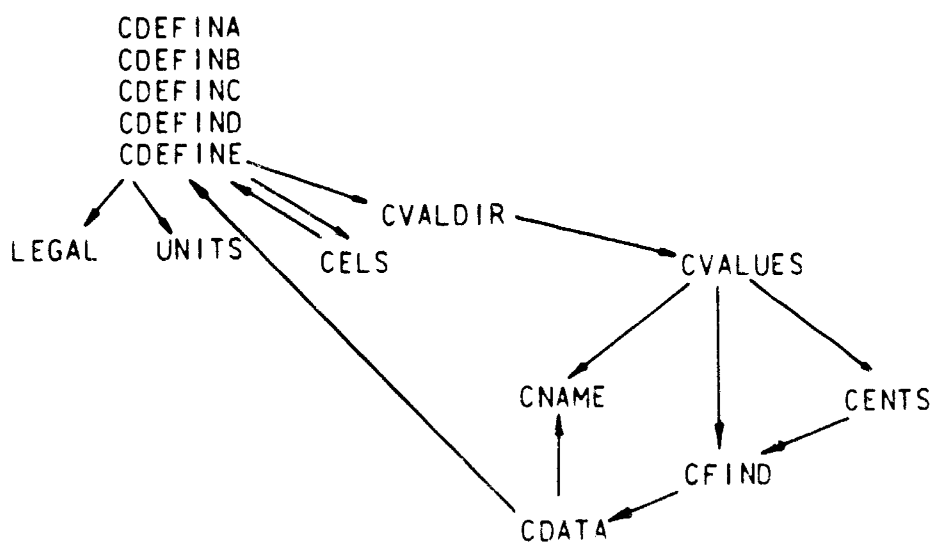


FIG. 1 - TDMS/RFMS TABLES

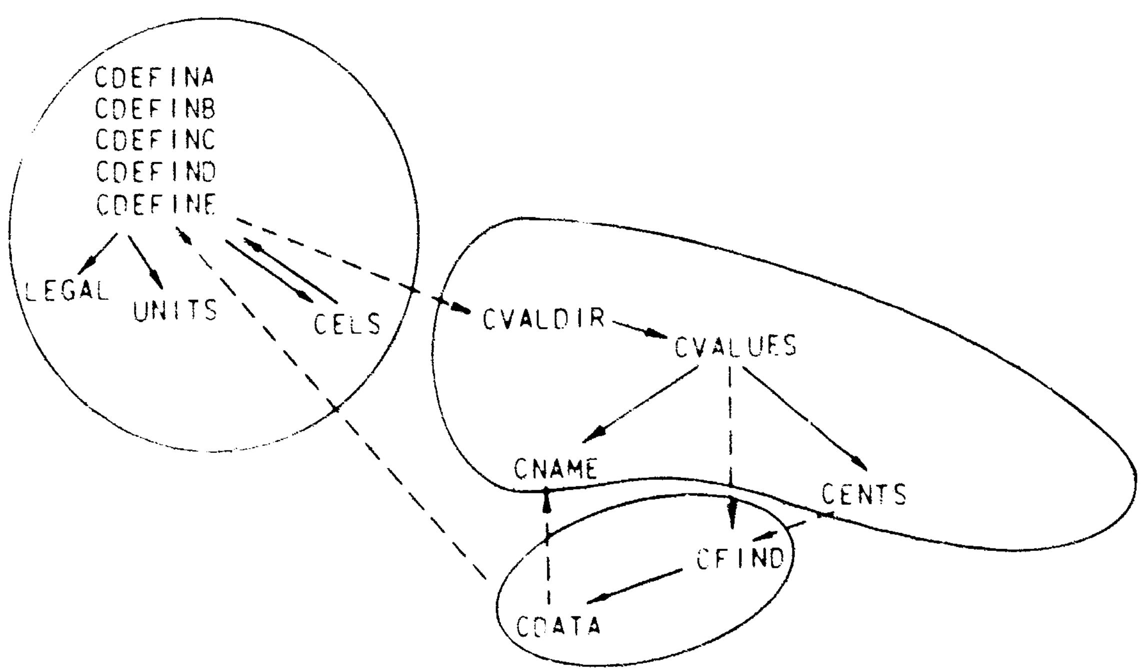


FIG. 2 - FUNCTIONAL GROUPS OF TDMS RFMS TABLES

TRACOR DWG A6-127-3
10/25/67 ZIEHE

2.1.2 Component Data

The eight tables of this group, CDEFINA, -B, -C, -D, -E, CELS, "legal", and "units", contain items of information about the components of a data base. Some of the items are meaningful for both elements and repeating groups, but many apply only to elements. Tables CDEFINA, -B, -C, -D, and -E have fixed-length entries, one for each component, stored in the order of definition. The system-assigned component number indexes each table. The other three tables have variable-length entries. CELS contains the alphanumeric name assigned each component. "Legal" and "units" have an entry for each element. An entry in the former will allow the system to make legality checks on values ascribed to an element. An entry in the latter will record the units of measurement for an element's values. Their contents have not been further defined at this time.

The following items apply to all components:

1. The component name assigned by the user with a measure of its length in both words and characters --
   a. a pointer to the name: table CDEFINA, item CPNAMEA
   b. the name: table CELS, item CARS
   c. the number of characters in the name: table CELS, item CNBCAR
   d. the number of words required for the name: table CELS, item CNBWDS
2. The component number assigned by the user -- table CDEFINB, item CFIELD
3. The component number assigned by the system -- table CELS, item CPEL (it serves as a pointer from the CELS table back to CDEFINA)
4. The component level number -- table CDEFINA, item CLEVEL

5. The component type tag -- table CDEFINA, item CTYPE

The following items are defined but have meaning only for elements:

1. Data which define the location of the value glossary for an element --
   a. a pointer to the CVALUES sub-table for this element: table CDEFINC, item CPVDIR
   b. the number of partitions required for that sub-table: table CDEFINC, item CVENT
   c. the base index for the CNAME sub-table for this element: table CDEFIND, items CNREL and CNLOC
   d. the base index for the CENTS sub-table for this element: table CDEFINE, items CEREL and CELOC
   e. the number of characters in the longest value assigned the element: table CDEFINA, item CMAX
2. Data which complete the definition of the component structure --
   a. repeating group for an element, element identification for a sub-element or overlay; table CDEFINA, item CRGID
   b. "parent", "sub-element", "other" tag: table CDEFINA, item CSPIN
3. Value descriptors --
   a. "value", "no value" tag: table CDEFINA, item CNOVAL
   b. type tag for numeric input: table CDEFINA, item CFLOAT

4. Pointer to the unit-of-measure -- table CDEFINA, it n CPUNIT

5. Pointer to the legality checks -- table CDEFINB, item CPLEGAL

2.1.3 Value Glossaries

The four tables in this group, CVALDIR, CVALUES, CNAME, and CENTS, contain glossaries of element-values, one for each element. Each is in a form convenient for searching, but in addition maintains connections between a value and the node or set of nodes in the datum array to which it corresponds. CVALDIR is a directory of the other sub-tables. CVALUES and CNAME contain the actual value glossaries. CENTS, and in some cases CVALUES also, maintains the connection between a value and nodes in the datum array.

Each sub-table of CVALUES and CNAME corresponds to a single element, and each entry corresponds to a distinct value. The CVALUES entry is fixed-length and accommodates some value descriptors. An item is also provided for the value or a truncated form of it. In the latter case the complete value is stored in a CNAME entry. The CVALUES entry also includes a pointer to the node in the array to which the value corresponds; if it corresponds to more than one, that pointer is directed to a sub-table of CENTS whose entries in turn point to each occurrence in the array.

2.1.4 Datum Arrays

The two tables of this group, CFIND and CDATA, record the node structure of a data base and relate each node to a value in the value glossaries. CFIND contains an entry for each set of values in the data base. In TDMS/RFMS a set is either the values on level 0 of a logical entry or the values in one instance of a repeating group.

Each CFIND entry contains two pointers to other CFIND entries; these pointers structure the table into a tree and allow movement up and down the hierarchy and through sibling sets. One pointer connects a set with its parent, if the set is not on level zero. If it is, it has no parent and the pointer connects the set to its next sibling set. The relationships established by these pointers are shown in Fig. 3.

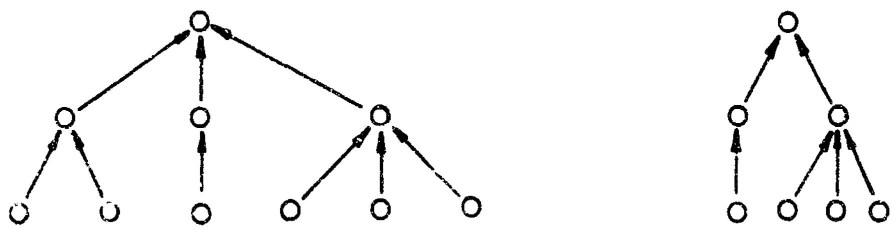
The other pointer connects a set to the next following set within the logical entry which is on the same level. For the last set on each level of an entry, this pointer is zero. The relationships established by these pointers are shown in Figure 4. Arrows represented by dotted lines are not pointers. Rather, they represent the fact that the first offspring of a set is always stored in CFIND as the next following entry.

Each CFIND entry also has a pointer to a sub-table of CDATA that contains an entry for each element-value of the set. Each sub-table entry is a pointer to an element-value as stored in the glossary of values for that element and the system assigned component number for the element.

In addition to the three pointers, each CFIND entry also contains the number of elements in the set it represents, the level number of the set, and the system's component number for the repeating group or logical entry of which this set is a part.

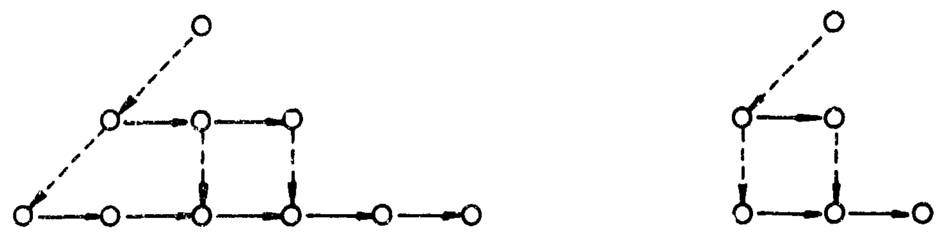
2.2 DM-1

The basic objective of DM-1 is to provide business organizations with a comprehensive data handling capability. It is assumed that the types of users will range from those who know nothing about the system and who wish to use it without learning more to those who understand the system well and who wish to manipulate its inner workings to their advantage. To this end the system has a convenient user-oriented set of languages.



THE UP POINTERS IN CFIND

FIG. 3



THE DOWN POINTERS IN CFIND

FIG. 4

TRACOR DWG A6-127-1
10/23/67 ZIEHE

Furthermore, all systems exist in an environment of change and those that can adapt to such changes continue to give useful service. Two adaptive aspects of DM-1 are its modular design, both with respect to its data base and program library, and its ability to restructure data, in response to a user's command as well as on the basis of statistics compiled by the system.

This description of objectives and approaches indicates a marked similarity to the goal set for TDMS and RFMS, but there is an important difference in the emphasis. DM-1 anticipates use by programmers as well as data-handlers and emphasizes the need for system adaptation to change.

DM-1 is designed for on-line operation as well as batch processing. Several users can work with the system concurrently, each with his own data base, but the system maintains all of the data under its control in a single data structure. Each user's data are incorporated into this structure, although each is able to interact with his array without consideration for other users or their data. All data, both descriptive and logical, are blocked and stored on secondary storage. No format distinction is made between storage on drum, disc, and tape. The system allows for matching the frequency-of-use with the access speed of the device. DM-1 and the operating system manage the traffic between the secondary and primary stores.

2.2.1 Internal Architecture of DM-1

All logical data associated with DM-1 datum arrays are stored in a permanent system structure formed by the rules outlined for DM-1 in Section 1. Each user's array of descriptive data is in a very real sense an extension of this structure. Therefore, in DM-1 both logical and descriptive data are stored in a single storage format. A single set of storage/retrieval operators applies to both. The system's architecture can be described in terms familiar to the user, enhancing his understanding

of system design and internal functioning. This approach also offers the system a measure of freedom for evolution instead of fixing its organization and content at an early design stage.

Perhaps the most significant effect of this policy is that it formalizes the system's design and regulates its implementation. Under this policy the list of exceptions and special cases that characterize much system documentation is controlled, if not eliminated. This is especially significant since, in software development, exceptions and special cases are often not documented completely. Of those not documented, some are remembered by the system implementors, but others are forgotten.

In DM-1 the Data Pool is the repository for all data, both logical and descriptive. It is a statement which subsumes four other statements -- the Data Base, the Directories, the Work Items, and the Scratch Area. The Data Base subsumes all user arrays given the system to manage. Its substructure is the combined structure of these arrays and, of course, changes as these arrays change or as they come and go. Work Items receives output from a job that is to serve as input for a subsequent one. After an item is used, it is discarded. Thus, this statement's substructure is a communications area and is controlled by the jobs using it. The content of the Scratch Area is even more temporary. Items are stored that are intermediate results, obtained during the course of executing a job. They must be used by the same job since they are destroyed when the job terminates.

The Directories statement subsumes all logical data for arrays under the system's control, that is, the content of the Data Base, the Work Items, and the Scratch Area. In addition, the Directories contain information required in two other system areas: program and job descriptions and user-related information including access and modification rights. The substructure of

the Directories represents the architecture of the system. Figure 5 shows the first level and its relationship to the rest of the Data Pool.

The fourteen sub-items that constitute the Directories can be grouped as follows. The first three, all files, contain logical data supplied by users for arrays being managed by the system. The next three files contain additional logical information collected by the system at a user's request and under his control. The next file is the control point for the system's internal segment storage and retrieval system. The next four items -- two statements and two files -- are the repository for the system's program and job descriptions. They are the interface between the datum arrays and the library of programs that process these data. The final three files contain information about system users and their rights relative to data in the Pool. In this paper we focus attention on only the first three groups of sub-items.

Section 1 defined three components or items for DM-1. A fourth, the record, is referred to although it is not used directly in structural descriptions. Two additional items are used in DM-1, the link item and the null node. The former is defined in Section ^.2.3. The latter is simply a place-holder; it has no sub-items and corresponds to no data. It is an artifice, used in file maintenance, to reserve a slot in the structure for subsequent redefinition.

One additional feature of DM-1 must be described prior to a closer look at the Directories. External names assigned the components identify each item of a structural description. However, records and links have no external names, and although external names are convenient for the user they are cumbersome for internal system use. Therefore, DM-1 uses a logical address for both components and data. External names are translated into logical addresses and vice versa.

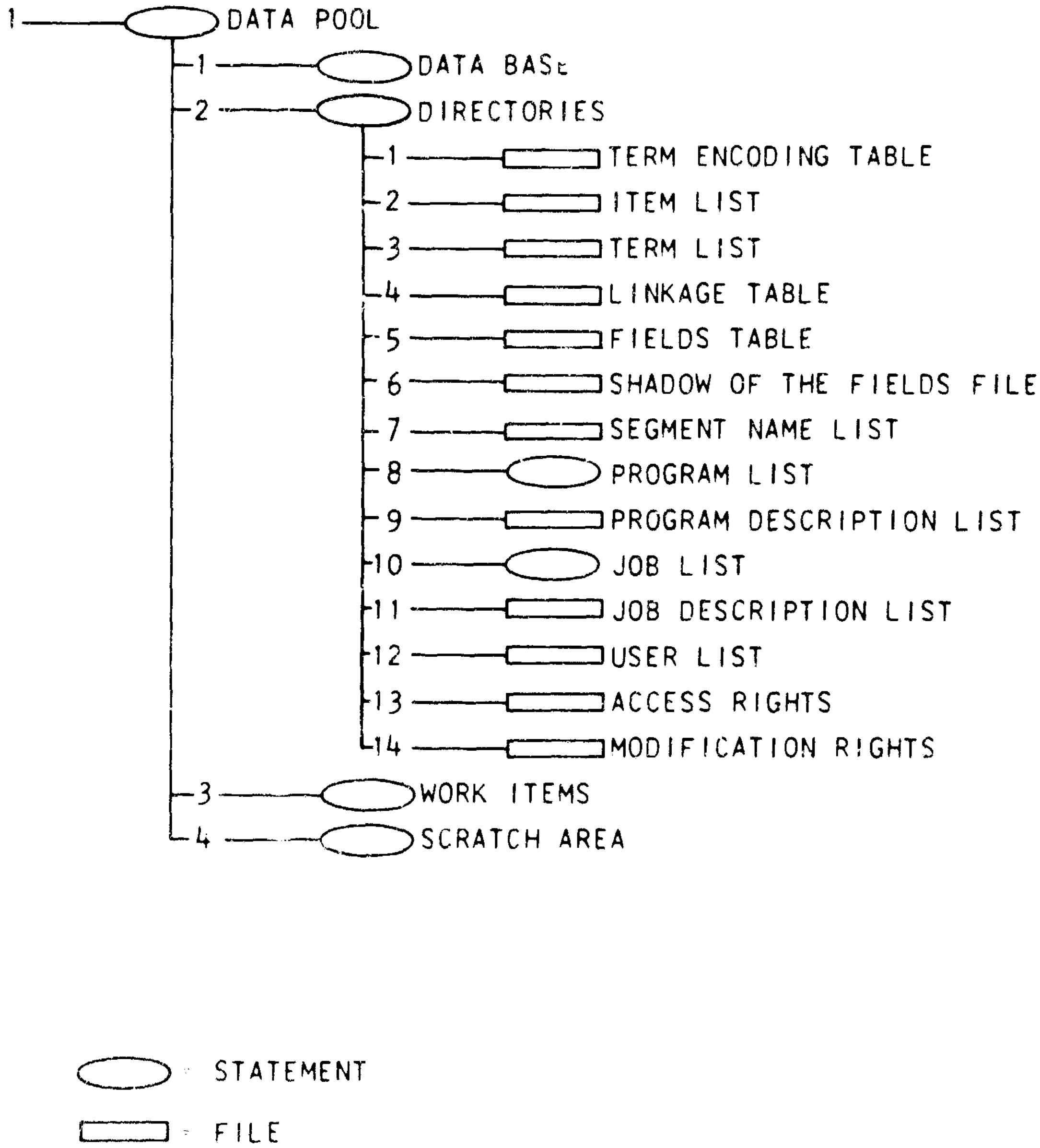


FIG. 5 - THE DM-1 DATA POOL - UPPER LEVELS OF THE STRUCTURE

TRACOR DWG A6 -127-2
10/25/67 ZIEME

A node's address contains one number for each hierarchical level; the sequence of numbers are separated by periods. The offspring of a node are numbered sequentially; this number, appended to the parent's logical address, is the logical address of an offspring. The logical address of an item carries the variable R in positions that correspond to record items and is called an Item Class Code (ICC). When all of the R's of an ICC have been replaced by record numbers, the logical address is an Item Position Code (IPC) and refers to a node in the datum array.

2.2.2 Logical Data from the User

The three files in this group are the Term Encoding Table, the Item List, and the Term List. When items (components) are defined, all of the information provided in the definition is stored in the latter two. The system duplicates the external names for items in the Term Encoding Table where they are stored in alphabetic order.

The Item List and the Term List are parallel files; each contains one record for each item defined and the records in both are ordered by the item's logical name, its ICC. The ICC is not explicitly stored in these files, but is instead either claculated or retrieved from the Term Encoding Table where it is stored.

The Item List records contain nine fixed-length fields, one of which is unassigned space. Two of the remaining eight, the Item Type and the Item Size, define the structure of the entire Data Pool. The type identifies each item as a statement, file, record, field, link (either source or target) or null node. For fields it further classifies items according to the encoding form of its values: floating point, integer, octal, decimal, B-S format, alphanumeric, or binary. The Item Size gives the number of items in the substructure of statements and files and the value size for fixed-length fields. If the values for a

field vary in length, the Item Size indicates that fact and each value is accompanied by a measure of its length.

The other six fields in Item List records contain a variety of item descriptors. Two specify the level of security restriction assigned to the item, one for access and one for modification. Two others indicate whether the system has collected additional logical data for the item and, if so, the record numbers in which it is recorded. They are discussed further in the next section. One field tags the item as optional or required and the final field specifies whether a definition has corresponding data or not.

The Term List records contain three fields, one of which is unused space. The other two contain the external name assigned the item and the units associated with its values. These fields are recorded in a separate file because the first is variable length and the other two are infrequently used.

The Term Encoding Table contains one record for each Term Name used for an item. Associated with each name is a file of ICC's, the ICC's for the items to which the name has been assigned.

2.2.3 Logical Data From the System

The DM-1 user can request that the system collect and store other logical data relevant to a datum array. Although the Data Pool is basically a tree structure, a user can introduce special items called links that relate separate branches of the tree. These connections are recorded in the Linkage Table. Similarly, although each datum value is stored in the Data Base according to its logical address (its IPC), the system can compile a glossary of values, called an index, together with the record number of each occurrence. These glossaries and record number files are stored in parallel files -- the Fields Table and the Shadow of the Fields File.

Link items are never part of the initial definition of a structure. The reason for this restriction is that item definition is by nature a sequential process, while linkages involve both a source and target item; one without the other is meaningless. Consequently, pairs of links are inserted into an established structural description. The source link, in some respects similar to a statement, is parent to the link criterion, a field whose value is the key to closing the link. The target link is parent to the criterion field in the target branch of the tree. When the values of the two criterion fields are equal, the source link logically subsumes the target link's parent item. In this respect the source link is like a file. Its records are the items of the target structure in which there is a match between source and target criterion values.

A DM-1 linkage is represented by records for the link items, both source and target, in the Item and Term Lists, and by two records in the Linkage Table. The record numbers of the records in the Linkage Table are the content of the Record Number fields in the Item List records, one for the source link and one for the target. Each Linkage Table record contains a source/target tag, the ICC of its partner link, and a field for a count of the times the link is traversed. Records are added to the Term List to keep it parallel with the Item List.

DM-1 stores values in the Data Base in order by logical address. This storage order makes no commitment to a particular type of further processing; it simply records user-defined relationships in minimal form. Values are preserved in this form as long as the array remains under system control. Any additions to the logical data or reorganizations of the descriptive data do not affect the Data Base in any way.

While the simplicity of this organizational form offers advantages, it is not well suited for non-sequential operations on the data. DM-1, therefore, offers a capability to index the

values of selected fields. Records in the Fields Table and its Shadow File in the Directories relate values of an indexed field to the records in which those values occur.

Three indexing modes are provided; all field values are indexed or indexing is restricted either to values on a list provided by the user or to values within a range prescribed by him.

The Fields Table file contains one record for each indexed field in the data base. When a field is indexed, the next available record number is assigned it and recorded in the Item List record for the field. There it serves as a link from the Item List to the Fields Table. The Fields Table record itself contains a tag that registers the indexing mode, a count of times the index is used, and a Field Value Table file. This embedded file contains one record for each value in the index. Associated with each value, but segregated into the separate Shadow File, is a file of record numbers for occurrences of that value in the Data Base.

2.2.4 Data Segments

All values for all fields that comprise the DM-1 Data Pool are stored in segments in secondary storage. A segment is fixed-length and serves as the unit of data transfer between DM-1 and the operating system. DM-1 documentation mentions a segment length of 9216 bits, but segments can, in principle, be any length.

Each segment consists of four parts: the head, the index, the body, and the slack. The head is fixed length, 96 bits, and consists of a segment name (54 bits), a pointer to the body (15 bits), a pointer to the slack (15 bits), and 12 unassigned bits.

The segment index, which begins with bit 97, is an extension of the Item List of the Directories. It is variable

length and contains logical data that pertain to a unique position in the array (an IPC) rather than to a class of items (ICC). The index contains record counters for files, present/absent flags for optional items, and lengths for variable-length values.

The body of the segment contains datum values as a stream of bits without field separators. Field boundaries are determined by values in either the Item List or the segment index. However, a null value bit accompanies each value. A system convention prevents the division of a value between segments. This limits a value to roughly the segment length minus 200 bits.

The slack of a segment includes all unused bits between the end of the body and the end of the segment.

A set of rules define the manner in which values are grouped into segments. For present purposes, it is sufficient to know that each segment is identified by the IPC of the first item which it contains. But since DM-1 expects the operating system to control relative and absolute mass-storage addressing, each segment is given a unique name by the operating system. This name is used by DM-1 in data transfer calls to the operating system. The Segment Name List cross-references the DM-1 segment identifiers (IPC's) and these segment names.

The Segment Name List is ordered by segment identifiers. Each record, in addition to an IPC, and the corresponding segment name, includes several fields for usage statistics and status information.

2.3 GIS

The documentation available for GIS is designed primarily for users of the system and therefore contains little information on the features being described in this section. However, some information is provided on data organization within a GIS file. We will limit the discussion in this section to that one feature of the system.

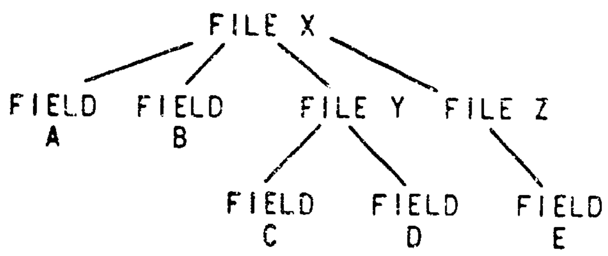
GIS provides three physical format options for files shown in Figure 6. In the linear record format the data in each record of a file is stored in the same sequence as its logical definition. This datum order corresponds to that used in DM-1 and catalogs.

In the second and third options, segments from each level of the hierarchy are stored separately. In the link-type split-record format, a link field, the last of each embedded segment, points to the parent segment on the next higher level. In the chain-type split-record format, each segment is linked to the first segment of each embedded file and to its next sibling segment. The last segment of a sibling set is linked to its parent segment.

2.4 CATALOGS

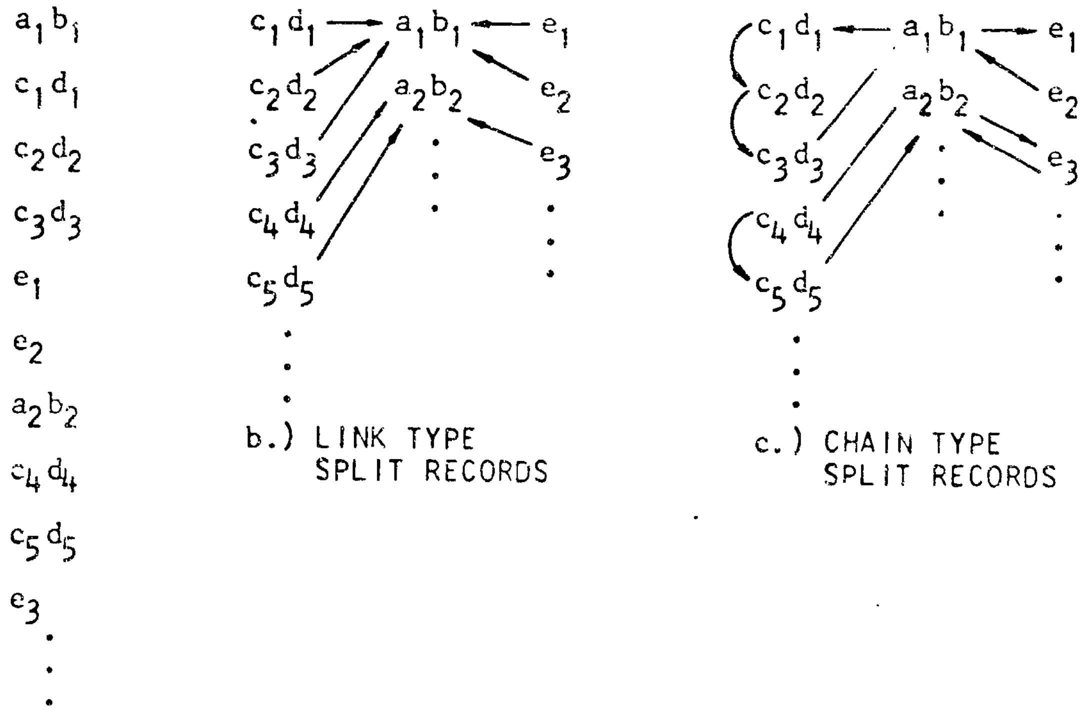
The Catalog system, as developed to date, is designed to facilitate long-term storage for files of structured data. Each catalog, both its map and the data that constitute its array, is written on a logical tape, one or more physical reels of magnetic tape. The map is the structural description in a fixed-format, one 36-bit machine word for each data class. The map and its corresponding data are blocked. Operations that block and unblock data as they move between core and tape are available in the system. The map of each active catalog is available in core and used by the system programs as they process data. Although concurrent operations are allowed on several catalogs, each is an entity, within core storage as well as on magnetic tape. Catalogs have been used in batch processing.

Two additional features of the catalog system are of interest here: the order in which data are recorded on tape and the tape format used. The datum ordering rule is the same as that used in DM-1 although in a catalog all data need not be at terminal nodes. No structural connections are made explicit,



A GIS STRUCTURE

***



a.) LINEAR RECORD FORMAT

b.) LINK TYPE SPLIT RECORDS

c.) CHAIN TYPE SPLIT RECORDS

FIG. 6 - GIS DATA STORAGE FORMATS

minimizing the logical data that must be recorded. No commitment is made to a particular type of further processing.

The blocking format used for catalogs mixes logical and descriptive data, in contrast to the format of segments in DM-1. The value of a catalog datum plus a tag that names its class forms a logical record. Logical records are packed into blocks and the length of each record recorded. The length of the logical record preserves the length of the datum value. Provision is made to continue a logical record from one block to another, thereby avoiding any limitation on the length of a value.

3. CONCLUSION

Sections 1 and 2 both point out differences among the systems discussed. This section simply collects these differences so that attention can be focused on them more easily.

Perhaps the basic difference in approach is the issue of system architecture: should it be constrained by formal rules, for example those made available to the user as in DM-1; or should it be a free form design as in TDMS/RFMS? The motivation for the latter approach is increased efficiency. The former approach opens doors to integrating all users' datum arrays with the system's data structure and to extending the system's data structure to include such information as validity checks, user access rights, and program and job descriptions. Some measure of machine independence is also won by machine independent rules for system design.

A second important area of difference is in the matter of system commitment to a particular type of future processing. DM-1, catalogs, and GIS each build much of the system's capability around data stored to directly reflect the user's structural description. TDMS and RFMS, however, are committed to retrieving data from a data base through use of element names and datum values. The datum arrays are stored in a form that expedites this activity, and other system capabilities reflect that organization.

Other differences lie in the area of rules for structuring data. Catalogs offer a single component for constructing structural descriptions while the other systems offer components with distinctive characteristics. Catalogs permit actual structuring of the data -- not all data are at terminal nodes; other systems group data under a hierarchical structure. DM-1 offers a device, the link node, for relating separate branches of the tree structure; the other systems do not. Finally, the rules

that govern the correspondence between map and datum array in each system except catalogs prevent any one-many or many-one relationships among the sibling data. Such relationships must be reflected in the hierarchical structure of the tree. This follows from the fact that repetitions are confined to files and repeating groups.

Subsequent work under this contract will incorporate features of these systems into a design for an item management capability suitable for semiotic systems. Process hierarchies as well as datum hierarchies characterize such systems. They also demand an open endedness which allows extension and adaptation, that is an evolutionary growth, as dictated by a particular application and the individuals which are a part of it. Differences of approach noted in this report will be resolved in the light of these considerations. The next report will sketch in broad outline a design for item management in semiotic systems; following that the details of the design will be developed and an operational version of the design will be implemented.

BIBLIOGRAPHY

1. The Time-Shared Data Management System: A New Approach to Data Management, by A. H. Vorhaus and R. D. Wills, SP-2747, System Development Corporation, Santa Monica, California, 13 February 1967.

2. Treating Hierarchical Data Structures in the SDC Time-Shared Data Management System (TDMS), by R. E. Bleier, SP-2750, System Development Corporation, Santa Monica, California, 29 August 1967.

3. Compose/Produce: A User-Oriented Report Generator Capability Within the SDC Time-Shared Data Management System, by W. D. Williams and P. R. Bartram, SP-2634, System Development Corporation, Santa Monica, California, 8 February 1967.

4. Working Documentation for Remote File Management System, by J. Morris, J. DeLine, and G. E. Autrey, Computation Center of The University of Texas, Austin, Texas, July 1967.

5. Reliability Central Automatic Data Processing Subsystem, Vol. I and Vol. II, Auerbach Corporation, September 1966, AD-489-666 and AD-489-667.

6. DM-1 -- A Generalized Data Management System, by P. J. Dixon and Dr. J. Sable, Auerbach Corporation, Proceedings of the Spring Joint Computer Conference, 1967, pp. 185-198.

7. Generalized Information System: Application Description, International Business Machines, E20-0179, 1966.

8. Reliability Central Automatic Data Processing Subsystem: Data Management System Survey, Vol. III, Auerbach Corporation, August 1966, AD-489-668.

9. The Catalog: A Flexible Data Structure for Magnetic Tape, by M. Kay and T. Ziehe, RM-4645-PR, The RAND Corporation, Santa Monica, California, October 1965.

10. The Catalog Input/Output System, by M. Kay, F. Valadez, and T. Ziehe, RM-4540-PR, The RAND Corporation, Santa Monica, California, March 1966.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D	
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)	
1. ORIGINATING ACTIVITY (Corporate author) TRACOR, Inc. 6500 Tracor Lane Austin, Texas 78731	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP
3. REPORT TITLE Data Management: A Comparison of System Features	
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)	
5. AUTHOR(S) (First name, middle initial, last name) Theodore W. Ziehe	

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 1967	40	10

8a. CONTRACT OR GRANT NO. N00014-67-C-0396 b. PROJECT NO. NR 048-239 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) TRACOR 67-904-U 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
10. DISTRIBUTION STATEMENT Distribution of the document is unlimited.	
11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Office of Naval Research Washington, D. C. 20360

13. ABSTRACT

Features of four data management systems under development are compared. The four systems are the Time-Shared Data Management System (System Development Corporation) and a variant of it, the Remote File Management System (Computation Center, The University of Texas); Data Manager - 1 (Auerbach Corporation); the Generalized Information System (IBM); and the Catalog System (The RAND Corporation). Comparisons are drawn in two areas: external and internal data structuring and organization. Several differences among the systems are noted and briefly discussed.

DD FORM 1 NOV 65 1473

Unclassified
Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Data Management						
TDMS						
RFMS						
GIS						
DM-1						
Catalog System						
Data Structures						

Unclassified
Security Classification